United States General Accounting Office

GAO

Report to the Chairman, Committee on Science, Space, and Technology, House of Representatives

June 1992

SPACE STATION

NASA's Software Development Approach Increases Safety and Cost Risks

GAO

United States
General Accounting Office
Washington, D.C. 20548

Jun 92

Information Management and Technology Division

Space Station: NASA's Software Development Approach Increases Safety and Cost Risks

B-247878

June 19, 1992

The Honorable George E. Brown, Jr.
Chairman, Committee on Science, Space, and Technology
House of Representatives

Dear Mr. Chairman:

This report responds to your request of April 9, 1991, for a study of software development issues for the space station. In this report, we address how well the National Aeronautics and Space Administration (NASA) has implemented key software engineering practices for the station: independent verification and validation, risk management, standards, and tools.

As requested, we did not provide a draft of this report to NASA for its review and comment. However, we discussed the report's contents with officials from NASA headquarters, the Space Station Program Office, and various work package centers. These comments have been incorporated into the report where appropriate. Our audit work was performed between April 1991 and May 1992, in accordance with generally accepted government auditing standards.

As agreed with your office, unless you publicly announce the contents of this report earlier, we plan no further distribution of it until 30 days from the date of this letter. We will then give copies to appropriate congressional committees; the Administrator, NASA; and other interested parties. Copies will also be made available to others upon request.

This work was performed under the direction of Samuel W. Bowlin, Director, Defense and Security Information Systems, who can be reached at (202) 512-6240. Other major contributors are listed in the appendix.

Sincerely yours,

Ralph V. Carlone

Ralph V. Carlone
Assistant Comptroller General

Executive Summary

Purpose

The National Aeronautics and Space Administration (NASA) is undertaking one of its most ambitious projects ever—Space Station Freedom. As part of this effort, NASA has started developing computer software that will drive the station's 10 main on-board computer systems. This software will perform critical functions ranging from keeping the station in its proper orbit to maintaining life support of the crew. In contrast with the space shuttle, software development for the station will be highly dispersed, with three prime contractors and scores of subcontractors across the country developing millions of lines of computer software code. This software is meant to last for the station's entire life—three decades.

Given the risks inherent in such a large, complex undertaking, the House Committee on Science, Space, and Technology asked GAO to determine (1) if independent verification and validation techniques are being used to ensure that critical software meets specified requirements and functions; (2) if NASA has incorporated software risk management techniques into the program; (3) whether standards are in place that will prescribe a disciplined, uniform approach to software development; and (4) if software support tools will help, as intended, to maximize efficiency in developing and maintaining the software.

Background

The space station is the linchpin of NASA's manned space program for the early 21st century. The 30-year project is an international venture involving cooperation with the Japanese, European, and Canadian space agencies. GAO has estimated that the space station will cost about $40 billion to develop, with total life-cycle costs of $118 billion.

The program has undergone a number of changes during the past several years that have affected many aspects of the station, including software development. In early 1993 NASA plans to complete critical design reviews – when key designs will be made final, and scores of geographically dispersed contractors will begin large-scale software development.

Results in Brief

While NASA plans to begin developing critical space station software soon, basic management control techniques that NASA and its contractors need to build and maintain high-quality software are not in place. As a result, safety and cost risks are increased. NASA has not implemented independent verification and validation of critical flight software and lacks a systematic approach to software risk management. In addition, NASA has been slow to

implement standards, has reduced funding for programwide support tools, and permits different software tool sets[1] to be used in different locations. The agency has not, however, assessed the long-term cost impact of these actions. Although NASA still has time to assess these issues, it is rapidly approaching a juncture of critical milestones, after which its ability to influence software development practices will be severely restricted—and much more expensive to correct.

Principal Findings

Failure to Implement Appropriate Controls Increases Safety, Cost Risks

Basic verification and validation (V&V) functions are normally performed by the builders of the software to help ensure that the software being developed meets requirements and performs as intended (see ch. 2). Independent V&V, however, seeks to attain an additional level of assurance whereby the products of the software development life cycle are independently reviewed, verified, and validated by an organization that is neither the developer nor the acquirer of the software. Government and industry guidelines strongly recommend that an independent agent be employed if failure of the software could result in loss of life or personal injury, mission failure, or catastrophic loss of property. Some space station software meets these criteria—particularly that controlling life-support systems and station operations. Nonetheless, NASA has not incorporated truly independent V&V into the program for its most critical software. What NASA labels as independent V&V is generally conducted by the same organization that builds the software and does not provide an added level of assurance over basic V&V activities. Program officials believe that little measurable value would be realized from using an independent V&V agent and that such a practice could be costly. For a critical and expensive software undertaking such as that for the space station, however, whether to employ independent V&V should not be based solely on the judgment of program officials without data and analysis of additional costs and risks.

Software risk management is another important management control mechanism needed for a project of this size and scope. It plays a pivotal role in avoiding safety and cost risks by taking preventive measures against them before they become sources of major rework. However, the station

[1]A tool set consists of hardware and software designed to enhance the productivity of software developers, including common editors, compilers, and other software utilities.

program lacks a systematic approach to software risk management. Due to this and other factors, longstanding software risk areas remain unresolved. For example, several years ago the program's engineering and integration contractor identified several high-risk areas—including the overutilization of computer processing and memory, and the failure to establish appropriate system redundancy—and recommended that NASA take action to address them. However, NASA continues to defer resolution of these issues. By ignoring software risks or addressing them late, serious safety risks remain unabated, and the cost of resolving these problems may be substantially higher than if they had been addressed early.

Decisions About Standards and Tools May Increase Life-cycle Software Costs

NASA's strategy for controlling software costs for the space station was to prescribe a software development methodology—a uniform, disciplined approach to software development—and provide a complete and consistent tool set to software developers. A prescribed software development methodology and a comprehensive tool set play a key role in facilitating integration and reducing long-term software maintenance costs. They also help NASA management exert control over a project of this size and scope. The methodology was to be provided by software standards and other methods and rules within a common environment for software development. This environment is known as the software support environment (SSE). The SSE was also to furnish developers with the tools needed for software development.

NASA's strategy for containing software costs has fallen well short of expectations. Because NASA failed to write software standards into contracts when contracts were awarded and has been slow to implement them subsequently, developers are writing flight software without needed software standards. In addition, because NASA developed the SSE concurrently with flight software, the SSE failed to support key development activities. As a result, integration and maintenance are likely to be more difficult; this may lead to higher software costs over the life of the program, schedule delay, or both.

With its original strategy for controlling software costs so badly weakened, NASA has chosen an approach that concentrates on minimizing short-term costs. For example, the agency has been hesitant to amend contracts to compel contractors to comply with many software standards because of increased short-term costs that result from such contract modifications. In addition, NASA has reduced funding for the SSE, and permitted and continues to permit different SSE tool sets to be used in different locations. The agency itself acknowledged the importance of software standards and

a common development environment in controlling software costs. Despite this, it plans to continue on its present course, without assessing how its failure to implement software standards or commit to a robust and uniform SSE could increase software costs over the life of the program.

Recommendations

In order to reduce safety and cost risks, GAO recommends that the Administrator, National Aeronautics and Space Administration, direct space station officials to (1) require independent verification and validation for critical space station software, and (2) institute a risk management program that identifies all key software risks and ensures that preventive measures are taken to minimize those risks.

To ensure that space station software is developed most efficiently, GAO also recommends that the Administrator direct space station officials to (1) perform a comprehensive evaluation comparing short- and long-term costs of implementing a prescribed software development methodology and fully supporting the program's software development environment, and (2) proceed in a manner consistent with the results of this evaluation. Such an evaluation should determine whether implementing software standards and committing to a robust and uniform software development environment will save money over the life of the program.

Agency Comments

As requested, GAO did not provide a draft of this report to NASA for its review and comment. However, GAO discussed the report's contents with the special assistant to the director of the space station program, as well as other senior officials at NASA headquarters, the space station program office, and the field centers. Their comments have been incorporated as appropriate. NASA officials at the field centers generally believe that the report is fair and accurately reflects current problems in the program. NASA officials at headquarters and at the space station program office believe that the agency has attended to the control techniques discussed in this report in the same way the agency has done business for past programs. These officials disagree that the approach to developing software that GAO described would help NASA develop safer or more economical software. However, the station's decentralized management structure, absence of a single prime contractor, and geographical dispersion of contractors all mark a significant departure from the way NASA has structured and managed programs in the past. In light of this, GAO believes it is all the more important that NASA implement the controls contained in this report for space station software development.

Contents

Figures

Abbreviations

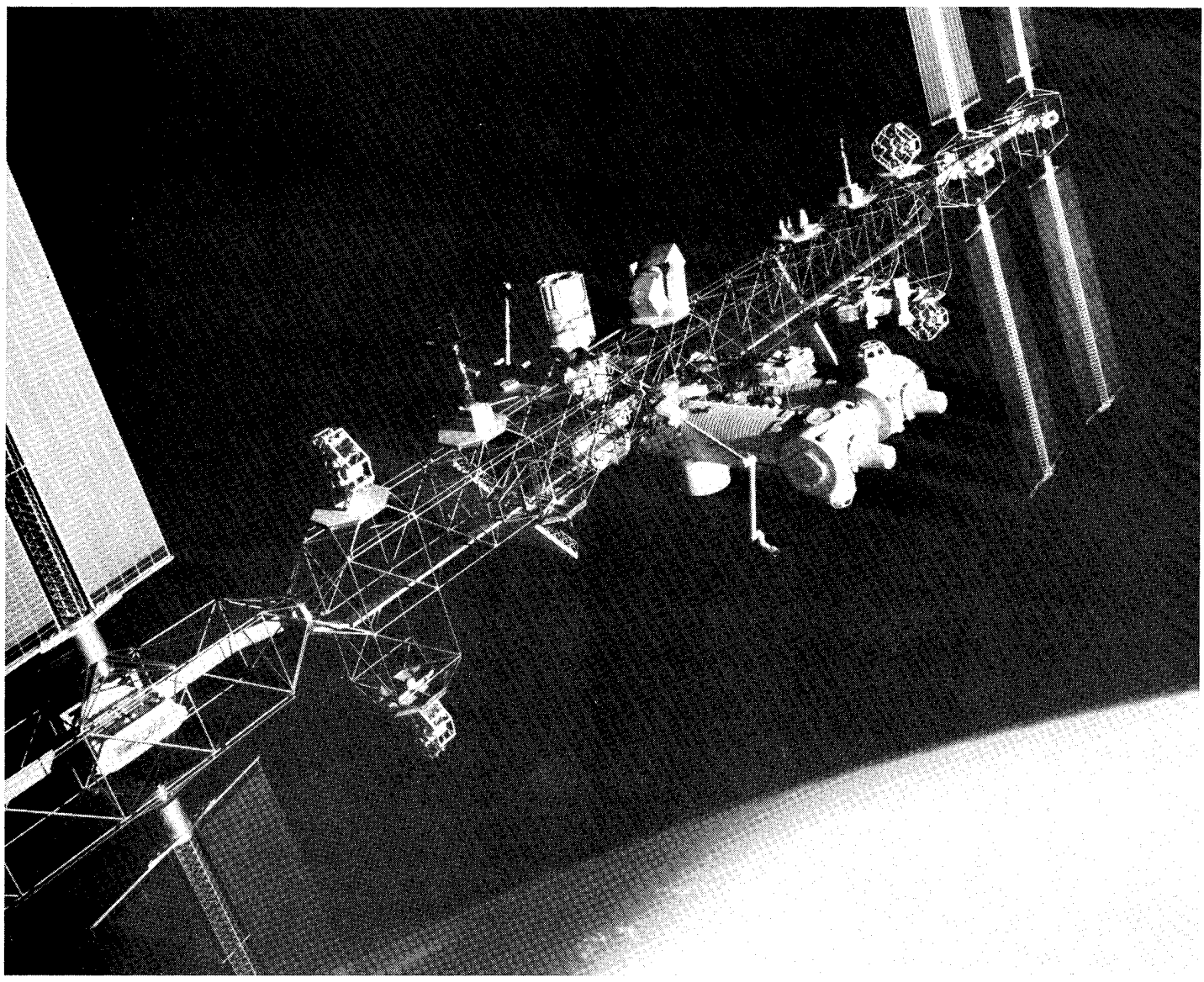
ANSI	American National Standards Institute
DMS	data management system
GAO	General Accounting Office
IBM	International Business Machines Corp.
IEEE	Institute of Electrical and Electronics Engineers
IMTEC	Information Management and Technology Division
NASA	National Aeronautics and Space Administration
NSIAD	National Security and International Affairs Division
SSE	software support environment
SSEIC	space station engineering and integration contractor
V&V	verification and validation

Introduction

The National Aeronautics and Space Administration (NASA) is undertaking one of its most ambitious projects ever—Space Station Freedom. The space station is the linchpin of NASA's manned space program for the early 21st century. The 30-year project is an international venture involving cooperation with the Japanese, European, and Canadian space agencies. The primary mission of the space station is to achieve U.S. preeminence in space exploration. Its scientific use is as a research laboratory to conduct microgravity and life-science experiments. As of last May, we estimated the space station project to cost about $40 billion to develop, with total life-cycle costs of $118 billion.[1] Figure 1.1 shows an artist's conception of the planned station.

[1]Questions Remain on the Costs, Uses, and Risks of the Redesigned Space Station (GAO/T-NSIAD-91-26, May 1, 1991).

Figure 1.1: Artist's Conception of the Planned Space Station



Source: NASA

Developing software for the station will be a major undertaking. Flight software for the station is expected to consist of over a million source lines of computer code, and the ground software will consist of millions more. This software will support the station's 10 main on-board computer systems, and will perform functions ranging from keeping the station in its proper orbit to maintaining life support of the crew. Table 1.1 illustrates the primary functions of each system. According to senior officials who have worked on other projects of this magnitude, software may turn out to be the single greatest cost item over the lifetime of the project.

Table 1.1: Space Station Systems and Primary Functions

Name of System	Primary Functions
Data Management	Data processing, command and control of other systems and payloads, and interface with the crew
Guidance, Navigation, and Control	Attitude control, rendezvous with space vehicles, and maneuvering of the station
Propulsion	Periodic reboost, attitude control, and collision-avoidance in conjunction with previous system
Communications and Tracking	Transmission of audio, video, operational, and experimental data within space and between space and ground
Environmental Control and Life Support	Control of temperature, humidity, air composition, and atmospheric pressure; control of water supply; processing and storage of human waste
Thermal Control	Maintaining equipment and structure within allowable temperature ranges
Electrical Power	Generation and distribution of power
Extravehicular Activity	Capability for pressure-suited crew members to operate outside of the pressurized main base
Fluid Management	Resupply and distribution of nitrogen and water, and disposal of waste gases
Manned Systems	Crew quarters, health care, food management, hygiene, housekeeping, and trash management

Program Management

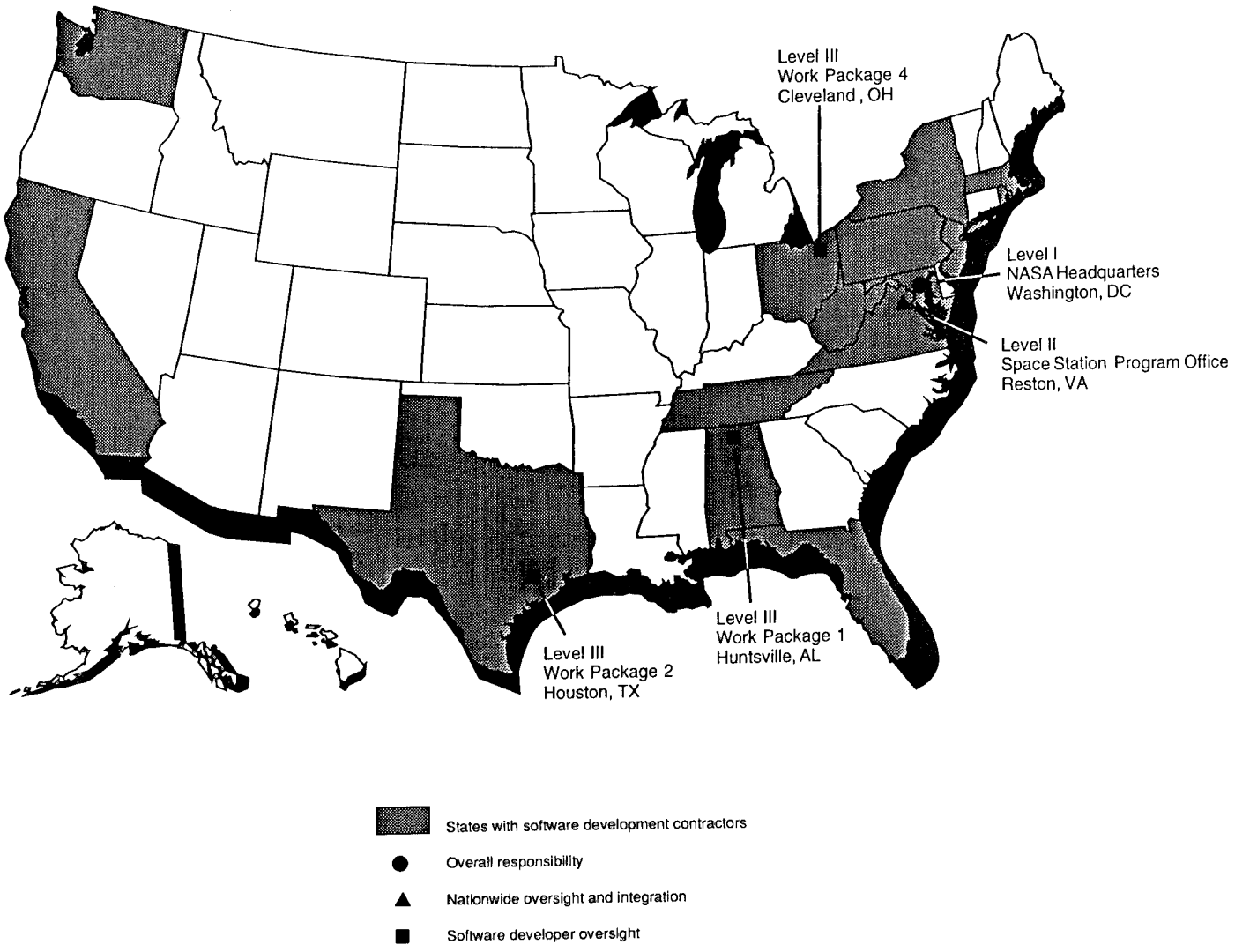
In contrast with software development for the space shuttle, which was centralized, software development for the station is highly dispersed, with three prime contractors and scores of subcontractors across the country developing software that NASA must integrate, maintain, and manage. This dispersed development environment greatly increases challenges to NASA in terms of managing this effort when compared with the effort for the shuttle.

No single prime contractor or NASA field center has overall responsibility and authority to manage software development for the station as part of this dispersed environment. This marks a departure from the way that many previous NASA programs were managed. Instead, NASA established what it termed work packages to develop the physical station, as well as software for the station. Each work package consists of a prime contractor, numerous subcontractors, and a NASA field center charged with principal oversight and management of that work package.

To manage the space station program, NASA instituted a management structure consisting of three principal tiers. These management tiers are known as Levels I, II, and III. Level I has overall responsibility for the program, and is located at NASA headquarters in Washington, D.C. Level II performs the bulk of nationwide oversight and integration, and is located mainly in Reston, Virginia. At this level NASA has also enlisted the support of a space station engineering and integration contractor (SSEIC), who performs responsibilities essentially in parallel with work done by Level II. Level III consists of three work package centers that oversee software development activity undertaken by the contractors who, in turn, deliver products to NASA.

Major systems are being designed and developed at the three work package centers: the Lyndon B. Johnson Space Center in Houston, Texas; the George C. Marshall Space Flight Center in Huntsville, Alabama; and the Lewis Research Center in Cleveland, Ohio. Each center manages one prime contractor. A space station projects office is located at each of the work package centers; this office includes a center software manager who reports to the center project manager. In turn, the center project manager reports to the overall program manager of the station. Figure 1.2 illustrates the geographical dispersion of software developers and the program management structure.

Figure 1.2: Software Development Community and Program Management Structure[a]



[a]Work Package 3 has been eliminated from the program.

History of Space Station Software Design and Development

Since 1984 the program has undergone major restructuring four times; each change has significantly altered software requirements, design, and development. As a result, the first launch is scheduled for late 1995, an 8-month delay, and the number of work package centers was reduced from four to three. Moreover, the software initiatives have occurred amid a period of constant management change. Turnover was frequent with NASA program managers and principal deputies. For example, the program has had five program managers with differing approaches and concepts.

As a result of past changes, NASA has not progressed much beyond the early stages of software design and development. NASA is currently involved principally in preliminary software design. During this phase, software system architectures and input/output interfaces are defined. Additional phases to follow in the program's 30-year life cycle are detailed design, implementation, systems testing, acceptance testing, and maintenance and operations.

Objectives, Scope, and Methodology

On April 9, 1991, the House Committee on Science, Space, and Technology asked us to study software development issues for the space station. We were asked to address how well NASA has implemented key software engineering practices for the station. Specifically, our objectives were to determine (1) if independent verification and validation techniques are being used to ensure that critical software meets specified requirements and functions; (2) if NASA has incorporated software risk management techniques into the program; (3) whether standards are in place that will prescribe a disciplined, uniform approach to software development; and (4) if software support tools will help, as intended, to maximize efficiency in developing and maintaining the software.

To meet our objectives, we

- reviewed and analyzed software development objectives and strategies contained in NASA conference publications;
- reviewed and analyzed NASA, other government, and industry guidelines for establishing good software development practices;
- reviewed and analyzed technical proposals and contracts;
- reviewed and analyzed software management plans, risk management plans, and program requirements;
- reviewed and analyzed reports prepared by NASA and contractor officials that identified key issues and challenges facing the program;

- obtained expert opinions on what constitutes appropriate independent V&V and software risk management activities;
- interviewed program officials at NASA headquarters in Washington, D.C.; at the Space Station Program Office in Reston, Virginia; and at the three work package centers: Johnson in Houston, Texas; Marshall in Huntsville, Alabama; and Lewis in Cleveland, Ohio; and
- interviewed contractor officials doing work for NASA at Johnson and Marshall.

Our audit work was performed in accordance with generally accepted government auditing standards, between April 1991 and May 1992. We discussed the contents of this report with senior NASA program officials and incorporated their comments where appropriate.

Appropriate Controls Over Software Development Are Lacking

Two management techniques key to controlling safety and cost risks associated with developing software for the space station are independent verification and validation (V&V) and a systematic approach to software risk management. However, NASA has not incorporated these techniques into the program. As a result, safety concerns about mission failure or loss of life due to a software failure are increased, as are concerns about higher long-term costs resulting from not implementing these mechanisms. Program officials appear unconvinced that investment now in these control techniques will yield the desired benefit over the life of the program. However, generally accepted precepts of software engineering indicate that they will; beyond this, in the interim, NASA runs largely unnecessary risks of increased safety hazards and long-term costs.

NASA Has Not Incorporated Independent Software Verification

Software V&V involves the analysis and testing of software throughout its life cycle to ensure that it meets requirements and performs as specified. Basic V&V activities—considered to be software assurance disciplines—are normally performed by the builders of the software. Independent V&V, however, seeks to attain an additional level of assurance whereby the products of the software development life cycle are independently reviewed, verified, and validated by an organization that is neither the developer nor the acquirer of the software and therefore has no stake in its success or failure. Government and industry guidelines strongly recommend that an independent agent be employed if failure of the software could result in loss of life or personal injury, mission failure, or catastrophic loss of property. Some space station software meets these criteria—particularly that controlling life-support systems and station operations.

NASA has not, however, incorporated independent V&V into the program for its most critical software. In general, NASA's actions fall short because a separate contractor is not employed to perform these quality assurance activities, and an added level of assurance over basic V&V is lacking. Despite our criticism in February 1991 of its failure to institute independent V&V for shuttle software,[1] NASA is taking a similar approach to station software. As a result, NASA runs the risk that performance problems and/or increased operational costs will result for station software.

[1]Space Shuttle: NASA Should Implement Independent Oversight of Software Development (GAO/IMTEC-91-20, Feb. 22, 1991).

Verification and Validation Help Produce High-quality Software

Software verification is "the process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase."[2] It usually involves reviewing, testing, and documenting that systems' requirements, design, code, and documentation conform to specified requirements. Verification leads to improvements in overall software quality and reduced costs by allowing early detection of errors and performance problems. Validation is "the process of evaluating software at the end of the software development process to ensure compliance with software requirements."[3] The difference between verification and validation is unimportant except to the theorist; practitioners use the term V&V to refer to all of the activities aimed at making sure the software will function as required.

As defined by government and industry standards, independent V&V is additional work above and beyond basic V&V normally performed by software developers. NASA software assurance guidelines recognize that an added level of assurance is inherent in independent V&V activities. According to a NASA guidebook, "the independent V&V activities duplicate the V&V activities step-by-step during the life cycle, with the exception that the independent V&V agent does no informal testing."[4] This work, however, must not substitute for the software developer's responsibility, but should complement and reinforce the developer's software engineering process, configuration management, and qualification test functions.

Independence, Added Level of Review Lacking

As described earlier, one of the conditions key to independent V&V is that it be performed by an organization that is neither the developer nor the acquirer of the software and therefore has no stake in its success or failure. However, this type of independence is generally lacking in the program. Only Marshall, which is responsible for about 18 percent of the software to be developed for the station, employs a truly independent contractor to perform V&V activities. The remaining 82 percent of software is to be "independently" verified by the same organization that develops the software, though by an organizational element that is—according to

[2]IEEE Standard Glossary of Software Engineering Terminology, Institute of Electrical and Electronics Engineers (IEEE), Inc., American National Standards Institute (ANSI), ANSI/IEEE Standard 729-1983, August 1983, p. 37.

[3]IEEE Standard Glossary of Software Engineering Terminology, p. 37.

[4]Software Assurance Guidebook, NASA-SMAP-GB-A201, September 1989.

NASA—technically and managerially separate from the element actually developing the software.

Another condition of independent V&V is that it provide an additional level of review over basic V&V activities. Basic V&V functions are normally performed by the builders of the software to help ensure that the software being developed meets requirements and performs as intended. With independent V&V, the reviewing agent provides another look at software that has already been reviewed by the software builder. However, in what NASA labels as independent V&V, the reviewing agent merely assists the software builder in the first look at the software being developed. As such, verification is done only once, and the added level of review intended to be provided by the V&V agent is lacking.

NASA Officials: Independent V&V Adds Little Value

Program officials are unconvinced that independent V&V as traditionally defined (i.e., done by a separate organization and providing an added level of assurance) adds much to the effort in terms of safety, quality assurance, or long-term economy. According to these officials, the additional costs and burdens incurred by independent V&V are clear, but its benefits are not. For instance, they stated that employing independent V&V can as much as double the cost of software development, yet few if any significant software mistakes may be identified. NASA officials also stated that checks and balances by independent agents are embedded within the software development process and that NASA's approach to software verification is based upon success that has been achieved in previous programs.

While it is true that the work of an independent agent requires an added level of review and increases short-term costs, such costs may pale in comparison with the price to be paid in terms of loss of life or property if critical software fails. Further, the costs of fixing errors late in the software development process because they went undetected earlier may exceed the cost of independent V&V. Despite previous successes, software development for the space station poses new risks due to the lack of a single prime contractor, the geographical distribution and large number of contractors, and the complexity of the software being developed. An independent second look at software would help to combat these risks. Finally, for a critical and expensive software undertaking such as that for the space station, whether to employ independent V&V should not be based solely on the judgment of program officials without data and analysis of additional costs and risks.

NASA Lacks Systematic Approach to Software Risk Management

Another important control mechanism needed for a project of this size and scope is software risk management. A software risk management program is the process of identifying, addressing, and reducing risks in the software development process before they become sources of additional cost. NASA, however, has yet to introduce a systematic approach to software risk management into the station program. Specifically, NASA is not using any programwide software risk management plan and treats software risk management unevenly across the program. As a result, key software risk issues are not being resolved. In addition, NASA is taking the chance that other serious risks may go undetected or, if detected, not be resolved in a timely fashion.

Software Risk Management Helps Produce Safe, Economical Software

A comprehensive software risk management system is important because it provides a continuous identification, assessment, resolution, and status check on a program's technical, schedule, and cost risks. In September 1987 McDonnell Douglas, the prime contractor for work at Johnson, wrote in a risk management plan[5] that a credible risk assessment approach starts with the evaluation of technical risks and integrates them with schedule and cost risks. This contractor developed a risk assessment model that was to be used throughout the life of the program to ensure that a status check is maintained on known risks on a continual basis and that new risks are identified in a timely manner, brought to management's attention, and assessed for potential impact and risk minimization approaches.

A key ingredient of a comprehensive software risk management system is a software risk management plan that lays out a consistent approach to dealing with software risks. In April 1989 a NASA software working group said that a software risk management plan should be developed and implemented for any critical software project.[6] The group described the purpose of this plan: "to assess software development risks and then control them through risk management planning, risk monitoring, and risk resolution."

Another reason that a systematic approach to software risk management is important is that it allows software risk issues to be tackled early—at lower cost—rather than dealing with such issues after they have become significant problems and sources of rework. Such rework includes

[5]Risk Management Plan, Work Package No. 2, Plan H4015, September 1987.

[6]NASA—Evolving to Ada: Five-year Plan, April 1989, p. 3.

additional analyses, redesigns, reprogramming, and retests, and can result in schedule delays, increased costs, and reduced functionality.

A September 1989 software risk management plan developed specifically for the station program provides statistics on why it is particularly important to limit the amount of rework performed in the software development process.[7] First, rework of software can consume up to 50 percent of the total cost of a software development project. Second, reworking a problem once software is in operation can cost up to 100 times the cost to fix the problem during the development of software requirements. Third, approximately 80 percent of the cost to fix software problems is spent on the top 20 percent highest risks.

Clear Plan, Consistent Treatment of Risk Issues Lacking

In June 1991 NASA introduced a draft version of a revised Software Management Plan that in a new, extensive section on risk management describes a consistent and comprehensive approach to software risk management. However, this plan has not yet been accepted as program policy. Thus, no clear programwide approach for identifying, categorizing, and reducing software risks exists. Moreover, the new section on software risk management is being introduced late in the process, after significant design and development work has taken place. As a result, program officials expect that contractors will be resistant to new requirements that are imposed on them for managing software risks, and that the section's content will be reduced before it is accepted as policy.

As long ago as September 1989, the station program commissioned a program subcontractor to develop such a software risk management plan.[8] Even though this plan was developed and workshops were held on how to apply material in the plan to the program, its use has been suspended since those workshops took place.

In the absence of a programwide plan, the program lacks a clear, cohesive strategy for managing software risks. While one of the program's participants developed such a strategy, it did so independent of direction by NASA and without any program requirement to do so. Without such a program requirement, program participants' approaches to software risk management are uneven across the program. For example, contractors for

[7]Software Risk Management Plan, TRW Huntsville Operations, Sept. 29, 1989.

[8]Software Risk Management Plan, TRW Huntsville Operations, Sept. 29, 1989.

work at Marshall and Lewis did not develop any plan comparable to the one previously mentioned for work at Johnson. In addition, the approach of contractors for work at Marshall and Lewis has been to identify general software risks and provide a status update on them, rather than establishing a thorough and sophisticated system for dealing with software risks, as the prime contractor for Johnson has done. Uneven treatment of software risk management across the program leaves NASA vulnerable to serious risks going undetected, or if detected, not being resolved in a timely fashion.

Level II, meanwhile, lacks the visibility needed to effectively manage work being conducted at the work packages. Level II's primary techniques for dealing with software risk issues are through a Program Software Control Board, which meets on a biweekly basis, and through periodic program reviews. However, rather than a careful process for identifying, categorizing, and reducing software risks, the Program Software Control Board is only a mechanism for agreeing upon and establishing program policy. In addition, periodic program reviews—sometimes a year or more apart—do not provide anything approaching the system for providing "continuous" identification, assessment, resolution, and status of the program's technical, schedule, and cost risks described earlier in the plan prepared by McDonnell Douglas. As such, neither the Program Software Control Board nor periodic program reviews provide Level II with a mechanism for regularly addressing software risks and ways in which work package centers are working to resolve them. This is necessary in order for Level II to provide direction, assign priorities, and allocate resources to reduce or resolve significant software risk issues.

Longstanding Risk Issues Remain Unresolved

Due to the lack of a systematic approach to risk management, serious risk areas that could have been identified and dealt with earlier in the program remain unresolved. These areas include (1) the failure to provide needed system redundancy, (2) inadequate processing and memory reserves, and (3) the lack of a stable software architecture.

In December 1989 questions were raised about not having adequate levels of hardware redundancy. In September 1990 the SSEIC reported that the station's data management system (DMS), which controls all other major systems, lacked the level of hardware redundancy needed to effectively control the risk of a critical failure. Even though the SSEIC said that the DMS needed to be able to withstand two hardware failures and still function, it is being built to withstand only one failure. Questions about needed

redundant levels of hardware also remain open for the systems that keep the station in its proper orbit and control temperature. Because the station will be occupied less than five percent of the time until the year 2000, the lack of sufficient hardware redundancy is especially dangerous. If the systems that control attitude fail during this time, it is unlikely that humans would be available to intervene. Consequently, the station could begin to tumble and possibly go into a hazardous or destructive orbit.

The station's on-board computers do not have an adequate level of processing and memory reserves. This increases the risk that NASA will underestimate the number of on-board processors required and that these processors will later be attempted to be utilized at levels above what they are able to supply. This risk was identified as early as September 1989. A year later, the SSEIC reported that NASA's processing and memory reserves were inadequate to accommodate station software, let alone allow for expected growth in processing and memory needs. Despite this finding, the problem remains unresolved.

The station does not yet have a stable software architecture. As a result, NASA runs the risk that system interdependencies will not be identified, requirements will not be met, and the systems will not perform as intended. Despite the identification of this risk in November 1990, NASA officials are still working on making software requirements final, getting better estimates of the number of lines of code necessary to fulfill these requirements, determining whether sufficient processor speed and memory will be available, and accommodating or modifying these requirements. Issues that need clearer resolution also include determining whether software will be written in Ada or a lower-level language, where specific software functions will reside, and how all system components will interface with each other.

NASA Has Not Assigned a Sufficiently High Priority to Software Risk Management

NASA officials stated that they are comfortable with the level of software risk management activity that has been incorporated into the space station program. In addition, some officials have asserted that explicit software risk management activities are not necessary if people in the space station program do their jobs correctly. According to the program's software engineering manager, the entire approach to program software engineering reduces risk.

These assertions, however, run contrary to a statement by NASA's Office of Safety and Mission Quality that "too often the truism that 'quality is

everybody's business' becomes 'quality is nobody's business' if specific responsibilities are not assigned."[9] Moreover, NASA's comfort with software risk management in the program as it currently exists is indicative of not assigning a sufficiently high priority to software risk management. Without a systematic approach to software risk management—including (1) a software risk management plan containing goals and strategies for identifying, categorizing, and reducing risks; and (2) a structured and consistent approach to software risk management across the program—major risks may not be identified. Even if they are, they may not be given the attention necessary to categorize and mitigate them effectively.

With the approach of critical design reviews in March 1993, NASA is rapidly approaching a crossroads where it will either implement independent V&V and a systematic approach to software risk management or leave itself vulnerable to increased risks of unsafe or diminished software performance, increased costs, and schedule delay. At the time of these reviews, major design decisions will be made final and scores of geographically-dispersed contractors will begin writing software on a large scale.

[9]Software Assurance Guidebook, SMAP-GB-A201, p. 5.

NASA's Cost-control Strategy Has Been Poorly Implemented

Key components of NASA's original strategy for controlling costs of space station software were to (1) prescribe programwide standards for key aspects of software development early in the program and (2) provide a complete and consistent tool set to be used by all software developers. However, with software development already underway at one site and about to begin on a large scale at the others, NASA has yet to fully implement either of these objectives. Because NASA failed to write software standards into contracts when contracts were awarded and has been slow to implement them subsequently, developers are writing flight software without needed software standards. Moreover, the program's software tool set is not yet complete, parts completed were available too late to support certain key development activities, and its use is not consistent across the program. As a result, NASA's strategy for controlling costs has been badly weakened. NASA is now proceeding with far less assurance that it can control software costs over the life of the program.

Original Cost-control Strategy Relied Upon Uniform Software Standards and Tools

NASA convened a panel of software development experts in 1984 to develop a strategy for effective management of space station software development. The panel recommended that a uniform software development methodology be implemented, including software standards and a robust tool set as the key elements. The panel noted the particular importance of standards for coding, documenting, and data naming, and urged that all standards be in place by the end of fiscal year 1985. The panel also emphasized the benefits of a common software development environment, both in saving time and money during the development phase and in enhancing NASA's ability to maintain the system over a long lifetime.

In 1986 the director of the space station program formally adopted the recommendations of this panel by promulgating as program policy the need to establish a uniform set of software standards as a component of a comprehensive software development environment.[1] This environment was named the software support environment (SSE).

[1]Level A Software Management Policies, Nov. 11, 1986, Sections 2.3 and 2.10, pp. 3, 5.

Key Standards Implemented Late—or Not at All

Key software standards were not in place prior to the issuance of requests for proposals and contract awards in 1988. Despite the aforementioned recommendation that all standards be in place by the end of fiscal year 1985, by 1986 NASA had only established policies specifying the need for software standards. Since that time, NASA has made slow progress in incorporating standards into contracts. As a result, contractors are now writing software without needed standards.

Several specific examples of standards that are still not implemented include: a software management plan, Ada coding standards, and a software metrics standard. By NASA's own definition, the software management plan is a key ingredient in defining programwide standards as requirements. Such a plan should contain the goals and objectives of the program; the technical and management approach to be employed; performance expectations, milestones, reporting requirements; and quality assurance procedures to follow. According to a 1984 NASA report, the lack of a programwide software management plan for the station program is a classic software management error that could contribute to a software disaster.[2] Lacking such a plan, NASA centers are following separate management plans developed by work package centers and software management forums.

In addition, the lack of standards for Ada coding is of immediate concern to several NASA officials. Without standards for Ada coding—standards needed to control the way Ada software is built—software may be more difficult and expensive to debug, integrate, and maintain. If implemented late in software design, these standards could result in increased software development and maintenance costs over the life of the program.

Finally, NASA has not yet adopted a standard for software metrics.[3] Currently, only guidelines exist for the reporting of software metrics. The lack of a uniform set of software metrics gives NASA program management inadequate visibility into all phases of the software development process—coding, testing, and integration. Without uniform methods for assessing software development progress, quality, and risk, NASA will have to rely upon a multitude of differing and possibly inconsistent methods in managing software development.

[2]Space Station Software Issues, NASA CP-2361, August 1984.

[3]In general, software metrics is the use of numerical methods for assessing software development progress, quality, and risk. For example, numerical methods are commonly used to estimate the number of source lines of code to be employed and number of programmers to be assigned.

NASA officials assert that some standards were in place at the time of contract awards in 1988, and that these standards have helped guide the work of software developers.[4] NASA officials also state that since the time of contract awards, they have made some progress in developing additional standards and making them binding on contractors.[5] The officials contend that even though it appears that standards are late, implementing standards now is appropriate because (1) they did not exist or were immature earlier in the program, and (2) a software design and development approach had not been selected prior to contract awards.

While NASA has made some progress in implementing standards since the time of contract awards, key standards in such areas as software planning, management, engineering, and security are still lacking. Without these key standards being implemented, management control over costs in both the development and maintenance phases remains weak. Finally, while NASA rightly points out that certain standards were immature at the time of contract awards, others were ready to be used and could have been implemented. These standards, as discussed, still have not been implemented.

NASA has been slow to implement standards in part because it is reluctant to increase program costs by amending existing contracts. In fact, NASA has established a program goal to avoid any costs associated with making standards binding in the near term. According to contractor officials, amending contracts to comply with many explicit standards now could have major cost repercussions. This is particularly true for standards that address software design and development activities that are already in progress. However, by avoiding any increase in near-term software costs, NASA is risking unknown and potentially much greater cost increases in the future.

[4]NASA's Information System Life-Cycle and Documentation Standards, version 3.0, and NASA's Level A Software Management Policies.

[5]These include standards that have been incorporated into programwide architectural control documents for the data management system and other major systems.

Poor SSE Implementation Has Eroded Its Goals

Another key part of NASA's strategy for controlling software costs—the software support environment—has been eroded by several factors, including alienation of users early on, late delivery of an incomplete environment to its users, and inconsistent use of the delivered portions of the environment across the program.

The SSE project's early activities consisted largely of having Lockheed, the prime contractor, enhance a proprietary software development environment owned by one of its subcontractors. However, this enhanced environment did not meet the needs of its users, and consequently, some users chose other tools. For example, according to a computer systems analyst from Boeing, for the first 1-1/2 to 2 years that the SSE was operational, it could be used only with great difficulty to generate documents, maintain document histories, and perform configuration management.[6] These shortcomings led NASA to incorporate commercially-available tools into the SSE. The key tool introduced was a commercially-available hardware and software package to assist primarily with the design and development of Ada code. This equipment, developed by a company called Rational, is now an integral part of the SSE. However, key developers had already selected alternative tools by the time the Rational equipment was added to the SSE and provided to them.

For example, although the technical proposal promised that the "SSE shall be developed by Lockheed with the clear understanding that the user's needs for tools to develop Data Management System software come first," the tools were not available when needed to support International Business Machines Corp. (IBM) in developing the DMS. DMS development comprises about 65 percent of all flight software development for the station. IBM needed SSE tool support by the spring of 1989 for early DMS design work. Because this support was unavailable, IBM used Rational equipment—though different models than are currently provided by the SSE—to help design and develop Ada code which, according to IBM, was done at minimal cost to the government. However, since IBM needs configuration management tools still not available on the SSE, it has selected its own configuration management system. NASA has incurred costs of about $5.4 million[7] to pay for IBM's cost of using this configuration management system and other tools still unavailable within the SSE.

[6]Configuration management is a process for maintaining and controlling changes to software requirements, specifications, code, and documentation.

[7]As of May 15, 1992.

Late delivery of an incomplete SSE has also led other contractors to seek out alternate software development tools. These developers have utilized in-house tools or obtained independent commercial support due to either inadequate or uncertain support from the SSE project. For example, to support early development work at Marshall, Boeing employed a system distinct from the SSE, for which it cites only a slight cost impact. In addition, due to uncertainty about what the SSE would provide, McDonnell Douglas, the prime contractor for work being conducted for Johnson, procured eight Rational machines. The cost to NASA for this purchase was approximately $3.5 million. The use of substitute tools by these and other developers undercuts the project's goals of containing costs and weakens the ability of the SSE to exert management control via the promised "complete and consistent support environment."[8] Erosion of the SSE and its goals is likely to make integration and maintenance more difficult. This, in turn, may lead to higher software costs over the life of the program, schedule delay, or both.

SSE Remains Incomplete

While the contract for the SSE preceded contracts for flight software design and development by about 14 months, this did not allow sufficient lead time to develop the tools and train users prior to the time the tools would be needed. According to a key developer, differences between what could be supplied and what was needed were so severe that had it not been for a congressionally-mandated restructuring of the space station, the SSE project might have been canceled. The restructuring postponed the needs of developers by several years. The SSE remains unfinished and is not expected to be completed until early 1993. As discussed, the DMS developer continues to use substitute tools to provide capabilities as yet unavailable on the SSE.

Unwillingness to Pay Now May Increase Life-cycle Software Costs

Having strayed from its original strategy for controlling software costs, NASA is now focusing primarily on minimizing its short-term costs. For example, the agency has been hesitant to amend contracts to compel contractors to comply with many explicit software standards because of the increased short-term costs that could result from such contract modifications. Program officials are currently pursuing the goal of avoiding any costs associated with making new standards contractually binding.

[8]Software Support Environment System Concept Document, LMSC F255415, October 1988, p. 3-1.

Also, of a planned $250 million budget, the SSE project office has cut $50 million. In doing so, the project office eliminated some capabilities of the SSE and services intended to be provided by the project office. For example, the SSE will not run on two of the four workstation types that the SSE was originally intended to support. Services the SSE project office is no longer going to provide include maintaining workstations, paying for some software licenses for commercial software packages, and offering user training and help.

In addition, NASA continues to permit different SSE tool sets to be used in different locations. Inconsistency among tool sets may result in integration and maintenance being harder, costing more, and taking longer than expected. NASA could have avoided these risks if it had fully developed the SSE and delivered the needed tools prior to beginning flight software development.

NASA's recent decisions about standards and tools fail to adequately consider the long-term impact on the program. While the agency initially acknowledged the importance of software standards and a common development environment in controlling software costs, it now acts without knowing whether its continuing failure to implement software standards or to commit to a robust and uniform SSE will increase software costs over the life of the program. With critical design reviews approaching in March 1993 NASA is approaching a point where it will either enforce discipline over the software development process or suffer heightened risks of increased life-cycle costs. At that time, key design decisions will be made final and scores of geographically-dispersed contractors will be given approval for large-scale software development.

Conclusions and Recommendations

NASA has not established management control techniques needed for a software development effort of this scope and complexity. It has not implemented independent V&V or a systematic approach to software risk management, and has made decisions about standards and tools without knowing how these decisions will affect long-term operations and maintenance costs. As a result, NASA makes itself vulnerable to serious safety and cost risks—risks that the proper application of these techniques would significantly reduce.

NASA's approach to independent V&V, software risk management, standards, and tools is short-sighted. By failing to implement needed control mechanisms and by showing only wavering support for mechanisms designed to decrease long-term costs, NASA has concentrated on decreasing software costs over the short-term rather than effectively controlling safety and cost risks over the life of the program.

If NASA fails to focus on long-term considerations, long-term safety and cost risks are likely to increase far beyond what NASA anticipates. Specifically, if independent V&V and a systematic approach to risk management are not implemented, serious and potentially catastrophic safety risks may go undetected or may be dealt with only after they have become critical, costly problems. In addition, by not adequately considering the impact of its continuing failure to implement key standards or commit to a robust and uniform software development environment, NASA runs the risk that software costs over the life of the program will be significantly higher than current estimates.

With critical design reviews approaching in early 1993—the time at which designs will be finalized and contractors will begin major software development activities—it is becoming increasingly important that NASA make an immediate and concerted effort to properly apply independent V&V, software risk management, standards, and tools across the program. By acting now, NASA can exert the management control needed to keep safety and cost risks from increasing far beyond what it anticipates.

Recommendations

In order to reduce safety and cost risks, we recommend that the Administrator, National Aeronautics and Space Administration, direct space station officials to (1) require independent verification and validation for critical space station software, and (2) institute a risk management program that identifies all key software risks and ensures that preventive measures are taken to minimize those risks.

To ensure that space station software is developed in the most efficient manner, we also recommend that the Administrator direct space station officials to (1) perform a comprehensive evaluation comparing short- and long-term costs of implementing a prescribed software development methodology and fully supporting the program's software development environment, and (2) proceed in a manner consistent with the results of this evaluation. Such an evaluation should determine whether implementing software standards and committing to a robust and uniform software development environment will save money over the life of the program.

Agency Comments

As requested, we did not provide a draft of this report to NASA for its review and comment. However, we discussed the report's contents with NASA officials, including the special assistant to the director of the space station program; the manager of the program engineering office; the manager of the utilization and operations office; the manager of the avionics systems office; as well as senior program officials at the Johnson Space Center and Marshall Space Flight Center. We have included their comments as appropriate. NASA officials at the field centers generally believe that the report is fair and accurately reflects current problems in the program. NASA officials at headquarters and at the space station program office believe that the agency has attended to the control techniques discussed in this report in the same way the agency has done business for past programs. These officials disagree that the approach to developing software that we described would help NASA develop safer or more economical software. However, the station's decentralized management structure, absence of a single prime contractor, and geographical dispersion of contractors all mark a significant departure from the way NASA has structured and managed programs in the past. In light of this, we believe it is all the more important that NASA implement the controls contained in this report for space station software development.

Major Contributors to This Report

Information Management and Technology Division, Washington, D.C.	Ronald W. Beers, Assistant Director John A. de Ferrari, Assignment Manager Gary R. Austin, Senior Computer Specialist Richard B. Weinstock, Evaluator-in-Charge Scott F. Robohn, Staff Evaluator
Dallas Regional Office	Merrie C. Nichols, Senior Evaluator Andy C. Clinton, Staff Evaluator

Related GAO Products

Questions Remain on the Costs, Uses, and Risks of the Redesigned Space Station (GAO/T-NSIAD-91-26, May 1, 1991).

Space Station: NASA's Search for Design, Cost, and Schedule Stability Continues (GAO/NSIAD-91-125, Mar. 1, 1991).

Space Shuttle: NASA Should Implement Independent Oversight of Software Development (GAO/IMTEC-91-20, Feb. 22, 1991).